Detection Solution

System Overview

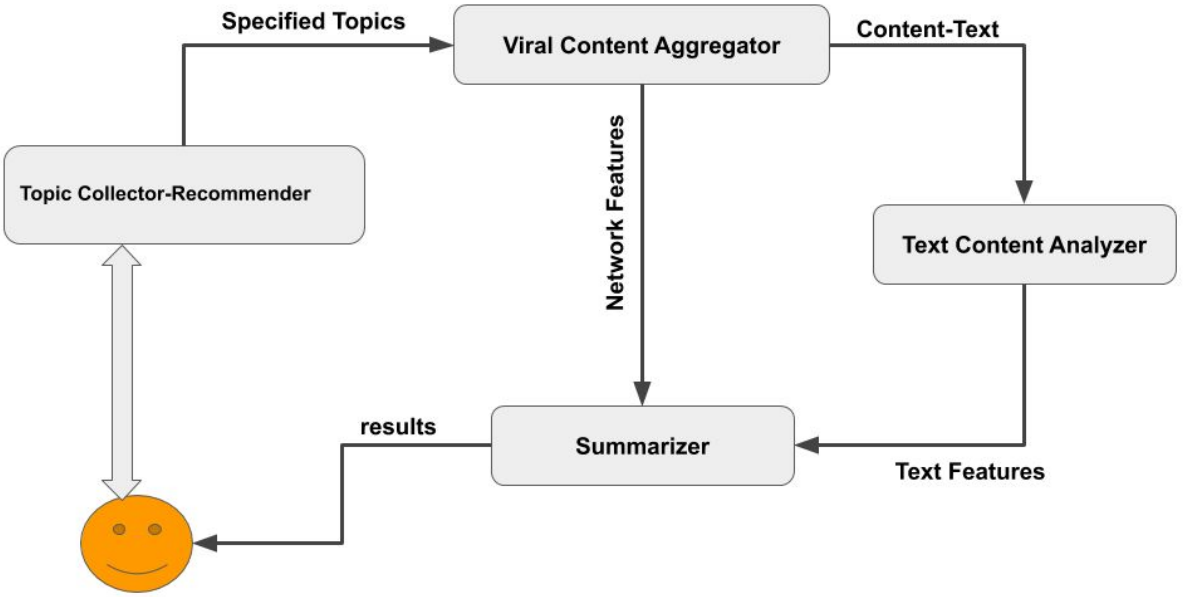
The system will leverage social/information network analysis and Natural Language Processing (NLP) principles to identify potential high risk information flow across different media platforms. The program consists of a core module which performs the detection task and a user interface to allow for easy interaction, visualization and analysis of the identified information. The system will be served with a web application as its user interface. Besides, an API (REST) access call can be made by other programs that might need the resources generated by the detection system. The system can be hosted in-house or we can leverage platforms such as the Amazon Web Service or Google Cloud. In house hosting is very expensive compared to cloud hosting services provided by Amazon Web Service and Google Cloud. Besides, the services provided by these platforms will enable the system to handle and process multiple requests and interaction in close to real time.

To perform a search, the user can provide a list of topics to consider in the analysis. If no list is provided by the user, a module of the system will recommend a list of trending topics that can be of interest to the user. The user will select the top n items from the recommended topics. The system will then search for media content related to the specified topics. An initial assessment/analysis will be performed to filter the identified contents based on an adjustable threat level threshold. The remaining data items deemed suspicious are passed to subsequent modules. The final stage returns the suspicious items ranked according to features including the threat-level, diffusion rate, stance etc. Overall the system will consist of modules for User Modelling, Propagation Modelling and the Content Modelling.

Required Skill

- Social Network Analysis
- Natural Language Processing
- Software Development and Engineering

System Architecture



The proposed system is visualized above. As shown it consists of four modules connected together for the detection task.. These are: Topic Collector-Recommender, Viral Content Aggregator (VCA), the Text Content Analyzer (TCA) and the Results Summarizer. As mentioned .

a. **Topic Collector-Recommender**: This module collects the topics provided by the user. However, if no topics are specified by the user, this module recommends a set of trending topics that might be of interest. The selected/specified topics are passed to the VCA.
b. **Viral Content Aggregator (VCA)**: This module scans different media platforms to identify contents related to the predefined topics. The module is particularly interested in the current trending discussions or posts, the information propagation graph and news articles related to the user specified topics. This would require deeper social media analysis. For each identified content, an algorithm will assign virality/visibility scores within the range [0-1] based on features/metrics including the number of shares and comments, origin/source of the content, references given and other information mined from the accounts of users who have read, commented or shared the post. Based on the visibility scores, the identified contents are divided into three groups: high-risk, medium-risk and low-risk contents. The items identified as low risk can be discarded. Only the high and medium risk contents are further analyzed. The output of the VCA is the network features employed to generate the visibility score of each identified suspicious content and the corresponding text-data.
c. **Text Content Analyzer (TCA):** using a set of NLP tools, this module will extract useful features from the given text. The tools employed include Named Entity Recognizer (NER) and the document embedding generator.The NER is leveraged by the **TCA** to recognize information units such as person, organization and location

names, numerical expressions including time, date, money. The document embedding generator is employed to generate the semantic representations of the given texts. The embedding generator can be a pre-trained language model such as BERT, ALBERT, GPT-2 etc. Alternatively with a large amount of data, we can efficiently train a custom language model. The text features generated from the embedding generator and the other nlp tools are passed to the **Summarizer**.

d. **Summarizer:** This is the core of the system making the final decision on the threat posed by the contents identified by the **VCA**. It takes as input the text features and the network features generated by the **TCA** and **VCA** respectively. These features are combined and used for the assessment task. The output of this module will include qualities such as the threat score within the range [0-1], probability that the content is fake or authentic, the stances of the contents in relation to the corresponding topics etc. All these will be supported with a fact checking module. Finally to explain why the system predicts an article or post as containing suspicious information, a sub-module exploiting the rich information within the social network or information diffusion is employed to identify user contents that supports the scores assigned. This sub-module is referred to as the **Explainer.** The **Explainer** exploits both the news/post contents and the users' comments to jointly capture the top-n sentences and comments that supports or counters the threat level assigned. These sentences and comments will help the user to understand predictions from the system.

System Development and Evaluation

The components of the system can be developed independently especially the **Text Content Analyzer** and **Viral Content Aggregator**. Some submodules within the **Summarizer** will need the outputs from the **VCA** and **TCA** modules during the development and deployment stages therefore it might be advisable that we build and develop them at the same time. For example, we can train the document embedding generator and the stance classifier in a multi-task learning style.

Some of the components require the training of deep neural networks. For example, the NER and document embedding generator. Deep learning is a data intensive task and as such to achieve higher performance, we will need a large amount of training data. The data can be obtained by scraping websites and other social media platforms (eg. Facebook and Twitter). Alternatively, we can fine-tune a pre-train model with a small amount of training data. Since the system will be handling user generated content online, there is a higher probability that the data is noisy. We will have to define an efficient preprocessing step to clean data. The annotation of the data set can be done using third party groups and companies including Amazon Mechanical Turks, and CrowdFlower. For the stance detection, we can use the SemEval-2016 Stance Dataset, RumourEval 2017,

PHEME collection of rumour and stance. The fact checking module will be trained in the FEVER: Fact Extraction and VERification dataset.

Primary, this system will be part of the backend to a web-application (user interface). The output of the **Summarizer** will be a structured data-frame with different fields and entries that can be converted into the figures and table on the user interface. The user interface will be a panel or a tab to display an overview of the user groups identified by the networks returned by the **VCA.** The user can zoom in and out to get different depth of information. Another panel will be a table and other visual contents (graphs) consisting of all the contents and their associated information returned by the **Summarizer.**

Overall, the performance of the system will be evaluated based on the number actual high/medium risk content identified. In addition, the model will be evaluated on how well it performs the information extraction and retrieval. When in deployment, the user can assign a performance score to the system after every request has been processed. To constantly improve the model's performance, we can either adopt the manual retraining of the model or continuous learning approach. Under the manual retraining (as the name implies) fresh data will have to be collected, re-train the model and re-deploy them. This is a very difficult task as we will have to decide when to perform the re-training, how often etc. As you can already imagine, this is a very difficult task. Therefore, we will employ the continuous learning approach. With this approach, we will have an automated system to constantly evaluate and retrain the models. Systems such as IBM Watson Data Platform with an in-built support for continuous training can be leveraged for this task.